Chapter 3 Objects and Classes
Topics

- OO Programming Concepts
- Creating Objects and Object Reference Variables
- Constructors
- Modifiers
- Instance and Class Variables and Methods
- Scope of Variables
An object

- Attribute 1
- ...
- Attribute n
- method 1
- ...
- method n

State

Behavior

A Circle object

- Attribute radius = 5
- Method findArea

OO Programming Concepts
Class and Objects

UML Graphical notation for classes

Circle
radius: double
findArea(): double

............ UML Graphical notation for fields

............ UML Graphical notation for methods

new Circle()
circle1: Circle
radius = 2

............ UML Graphical notation for objects
circle2: Circle
radius = 5

......

new Circle()
circlen: Circle
radius = 5
class Circle {
    double radius = 1.0;

    double findArea() {
        return radius * radius * 3.14159;
    }
}
Declaring Object Reference Variables

ClassName objectReference;

Example:
Circle myCircle;
Creating Objects

objectReference = new ClassName();

Example:

myCircle = new Circle();

The object reference is assigned to the object reference variable.
Declaring/Creating Objects in a Single Step

```java
ClassName objectReference = new ClassName();

Example:
Circle myCircle = new Circle();
```
Differences between variables of primitive Data types and object types

- **Primitive type**
  - `int i = 1`
  - `i: 1`

- **Object type**
  - `Circle c`
  - `c: reference`

Created using `new Circle()`
Copying Variables of Primitive Data Types and Object Types

- **Primitive type assignment**
  - Before: \( i \) = \( 1 \)
  - After: \( i \) = \( 2 \)

- **Object type assignment**
  - Before: \( c1 \)
  - After: \( c1 \)

\( c2 \): Circle
\( \text{radius} = 9 \)
Garbage Collection

As shown in the previous figure, after the assignment statement `c1 = c2`, `c1` points to the same object referenced by `c2`. The object previously referenced by `c1` is no longer useful. This object is known as garbage. Garbage is automatically collected by JVM.
TIP: If you know that an object is no longer needed, you can explicitly assign null to a reference variable for the object. The Java VM will automatically collect the space if the object is not referenced by any variable.
Accessing Objects

- Referencing the object's data:
  
  `objectReference.data`
  `myCircle.radius`

- Invoking the object's method:
  
  `objectReference.method`
  `myCircle.findArea()`
Constructors

Circle(double r) {
    radius = r;
}

Circle() {
    radius = 1.0;
}

myCircle = new Circle(5.0);

Constructors are a special kind of methods that are invoked to construct objects.
A constructor with no parameters is referred to as a default constructor.

- Constructors must have the same name as the class itself.
- Constructors do not have a return type—not even void.
- Constructors are invoked using the new operator when an object is created. Constructors play the role of initializing objects.
Visibility Modifiers and Accessor Methods

By default, the class, variable, or data can be accessed by any class in the same package.

- **public**
  The class, data, or method is visible to any class in any package.

- **private**
  The data or methods can be accessed only by the declaring class.

The get and set methods are used to read and modify private properties.
Passing Objects to Methods

- **main method**
  - `n = 5`
  - `myCircle = Reference`

- **printAreas method**
  - `c = Reference`
  - `myCircle = Reference`
  - `myCircle: Circle`
    - `radius = 1`
  - Pass by value (here the value is 5)
  - Pass by value (here the value is the reference for the object)
Instance Variables, and Methods

- Instance variables belong to a specific instance.

  Instance methods are invoked by an instance of the class.
Class Variables, Constants, and Methods

- Class variables are shared by all the instances of the class.
- Class methods are not tied to a specific object.
- Class constants are final variables shared by all the instances of the class.
To declare class variables, constants, and methods, use the static modifier.
Class Variables, Constants, and Methods, cont.

UML Notation:
+ : public variables or methods
- : private variables or methods
underline: static variables or methods

radius is an instance variable, and
numOfObjects is a class variable

```
CircleWithStaticVariable

+radius
-numOfObjects

+getRadius(): double
+setRadius(radius: double): void
+getNumOfObjects(): int
+findArea(): double

radius

circle1:Circle

-radius = 1
-numOfObjects = 2

instantiate

circle2:Circle

-radius = 5
-numOfObjects = 2
```

Memory

```
Memory

1
radius

2
numOfObjects

5
radius
```
The scope of instance and class variables is the entire class. They can be declared anywhere inside a class.

The scope of a local variable starts from its declaration and continues to the end of the block that contains the variable. A local variable must be declared before it can be used.